COMSC-210 Lab 16
Graph Routes

GOOD PROGRAMMING PRACTICES *show / hide*

ABOUT THIS ASSIGNMENT
In this lab assignment you will develop a graph-based solution for a roadmap, the data for which is stored in a text file of city names and distances between adjacent cities. In the first lab you will use shortest-route algorithm to find routes between 2 user-specified cities. In the second lab you will use cheapest-route algorithm to find routes between 2 user-specified cities.

Abstract data types will *not* be used. For the route calculation you will use the STL list template for the adjacency lists. We'll also make use of the STL queue, the STL stack, and the STL pair.

After you complete this lab assignment, post the required files to the COMSC server so that you receive the proper credit for this 50-point lab. Your posted files are due at midnight of the evening of the due date indicated in the course outline. Use the "Submit your FA2015 work" link on the class website to post your file for this lab, using the **lab16** folder provided.

LAB 16a: Write A Function For Shortest Route `[ GraphShortest.cpp ]`
Write **GraphShortest.cpp**, by completing the supplied CPP file. Right-click here to download the supplied file that contains the main function and the function for shortest route. Right-click here to download the input file to use, containing road map information for California. Write the body for the missing shortest route function, based on the algorithm in the lecture notes.

Note that the adjacency list is STL `pairs` of ints and doubles. Traverse it with a `list<pair<int, double> >::iterator`. Then use the `.first` and `.second` public data members of the `pair` to access the neighbor's index and the cost of getting there.

Submit the CPP to the class website for credit. Do NOT submit the TXT file.

Program I/O. Input: As prompted, a start city and a destination city. Output: The shortest calculated routes from start to destination.

Program I/O. Here's what the result should be for Coos Bay to Yuma:

```
Enter the source city [blank to exit]: CoosBay
Enter the destination city [blank to exit]: Yuma
Total edges: 5-CoosBay-Eureka-SanFrancisco-Bakersfield-LosAngeles-Yuma
```

LAB 16b: Write A Function For Cheapest Route `[ GraphCheapest.cpp ]`
Write **GraphCheapest.cpp**, by modifying the CPP file for the shortest route. Modify your CPP from 16b, replacing the shortest route function with a cheapest route function, using Dykstra's algorithm. Add any struct definition(s) as needed.

Submit the CPP to the class website for credit. Do NOT submit the TXT file.

Program I/O. Input: Same as 16b Output: Same as 16b, except for the cheapest route, with #of miles outputted instead of #of edges

Program I/O. Here's what the result should be for San Francisco to Reno:

```
Enter the source city [blank to exit]: SanFrancisco
Enter the destination city [blank to exit]: Reno
Total miles: 229-SanFrancisco-Sacramento-Reno

Enter the source city [blank to exit]: Yuma
Enter the destination city [blank to exit]: CoosBay
Total miles: 1164-Yuma-LosAngeles-Bakersfield-SanFrancisco-Eureka-CoosBay
```

How to pause a console program: *show / hide*

GRADING RUBRIC *show/hide*
